Class 23: Universal Machines

Schedule

Problem Set 9 is tomorrow, **Wednesday, 23 November** at **6:29pm**.

Problem Set Omega is now posted and due on **Sunday, 4 December** or **Tuesday, 6 December** (see problem set for details). It is not like the others, and counts as a "bonus" optional assignment.

Turing Machine

$$TM = (S, T \subseteq S \times \Gamma \rightarrow S \times \Gamma \times \textit{dir}, q_0 \in S, q_{Accept} \subseteq S)$$

S is a finite set (the "in-the-head" processing states)
Γ is a finite set (symbols that can be writte on the tape)
$\textit{dir} = \{\textbf{Left}, \textbf{Right}, \textbf{Halt}\}$ is the direction to move on the tape.

An *execution* of a Turing Machine, $TM = (S, T \subseteq S \times \Gamma \rightarrow S \times \Gamma \times \textit{dir}, q_0 \in S, q_{Accept} \subseteq S)$, is a (possibly infinite) sequence of **configurations**, $(x_0, x_1, \ldots)$ where $x_i \in \textit{Tsil} \times S \times \textit{List}$ (elements of the lists are in the finite set of symbols, Γ), such that:

```
- $x_0 = (\text{\bf null}, q_0, \text{\bf input})$
- and all transitions follow the rules (need to be specified in detail).
```

Machines and Languages

A **language** is a (possibly infinite) set of *finite* strings.

Σ = alphabet, a finite set of symbols
$L \subseteq \Sigma^*$

$$L_{XOR} = \{x_0 x_1 \ldots x_n + y_0 y_1 \ldots y_n = z_0 z_1 \ldots z_n \$ | x_i \in \{0,1\}, y_i \in \{0,1\}, z_i = x_i \oplus y_i\}$$

Turing Machine Computation

A Turing Machine, $M = (S, T \subseteq S \times \Gamma \rightarrow S \times \Gamma \times \textit{dir}, q_0 \in S, q_{Accept} \subseteq S)$, **accepts** a string *x*, if there is an execution of *M* that starts in configuration $(\textbf{null}, q_0, x)$, and terminates in a configuration, (l, q_f, r), where $q_f \in q_{Accept}$.

A Turing Machine, $M = (S, T \subseteq S \times \Gamma \rightarrow S \times \Gamma \times \textit{dir}, q_0 \in S, q_{Accept} \subseteq S)$, **recognizes** a language L, if for all strings $s \in L$, M accepts s, and there is no string $t \notin L$ such that M accepts t.

```
TuringMachine = namedtuple('TuringMachine', ['rules', 'q_0', 'q_accepting'])

def simulate(tm, starttape):
    tape = starttape
    headpos = 0
    currentstate = tm.q_0

    while True:
        readsym = tape[headpos]
        if (currentstate, readsym) in tm.rules:
            (nextstate, writesym, dir) = tm.rules[(currentstate, readsym)]
            tape[headpos] = writesym
            currentstate = nextstate
        else:
            return currentstate in tm.q_accepting # no rule, halt

        if dir == 'L':
            headpos = max(headpos - 1, 0)
        elif dir == 'R':
            headpos += 1
            if len(tape) <= headpos: tape.append('_') # blank
        else: # assert dir == 'Halt'
            return currentstate in tm.q_accepting

xorm = TuringMachine(
    rules = { ('S', '0'): ('R0', '-', 'R'), ('S', '1'): ('R1', '-', 'R'),
              ('S', '+'): ('C', '+', 'R'),
              ('R0', '0'): ('R0', '0', 'R'), ('R0', '1'): ('R0', '1', 'R'),
              ('R1', '0'): ('R1', '0', 'R'), ('R1', '1'): ('R1', '1', 'R'),
              ('R0', '+'): ('X0', '+', 'R'), ('R1', '+'): ('X1', '+', 'R'),
              ('X0', 'X'): ('X0', 'X', 'R'), ('X1', 'X'): ('X1', 'X', 'R'),
              ('X0', '0'): ('Y0', 'X', 'R'), ('X0', '1'): ('Y1', 'X', 'R'),
              ('X1', '0'): ('Y1', 'X', 'R'), ('X1', '1'): ('Y0', 'X', 'R'),
              ('Y0', '0'): ('Y0', '0', 'R'), ('Y0', '1'): ('Y0', '1', 'R'),
              ('Y1', '0'): ('Y1', '0', 'R'), ('Y1', '1'): ('Y1', '1', 'R'),
              ('Y0', '='): ('Z0', '=', 'R'), ('Y1', '='): ('Z1', '=', 'R'),
              ('Z0', 'X'): ('Z0', 'X', 'R'), ('Z1', 'X'): ('Z1', 'X', 'R'),
              ('Z0', '0'): ('B', 'X', 'L'),  ('Z1', '1'): ('B', 'X', 'L'),
              ('B', '0'): ('B', '0', 'L'),   ('B', '1'): ('B', '1', 'L'),
              ('B', '+'): ('B', '+', 'L'),   ('B', 'X'): ('B', 'X', 'L'),
              ('B', '='): ('B', '=', 'L'),   ('B', '-'): ('S', '-', 'R'),
              ('C', 'X'): ('C', 'X', 'R'),   ('C', '='): ('C', '=', 'R'),
              ('C', '$'): ('Accept', '$', 'Halt') },
    q_0 = 'S', q_accepting = { 'Accept' })

simulate(xor_machine, list("00+10=10$"))
```

